GeoInformatica
https://doi.org/10.1007/s10707-024-00528-6

RESEARCH

CLMTR: a generic framework for contrastive multi-modal trajectory representation learning

Anqi Liang[1] · Bin Yao[1] · Jiong Xie[2] · Wenli Zheng[1] · Yanyan Shen[1] · Qiqi Ge[1]

Received: 25 March 2024 / Revised: 12 August 2024 / Accepted: 20 August 2024

Abstract
Multi-modal trajectory representation learning aims to convert raw trajectories into low-dimensional embeddings to facilitate downstream trajectory analysis tasks. However, existing methods focus on spatio-temporal trajectories and often neglect additional modal features such as textual or imagery data. Moreover, these methods do not fully consider the correlations among different modal features and the relationships among trajectories, thus hindering the generation of generic and semantically enriched representations. To address these limitations, we propose a generic Contrastive Learning-based Multi-modal Trajectory Representation framework, termed CLMTR. Specifically, we incorporate intra- and inter-trajectory contrastive learning components to capture the correlations among diverse modal features and the intricate relationships among trajectories, obtaining generic and semantically enriched trajectory representations. We develop multi-modal feature embedding and attention-based fusion approaches to capture the multi-modal characteristics and adaptively obtain the unified embeddings. Experimental results on two real-world datasets demonstrate the superior performance of CLMTR over state-of-the-art methods in three downstream tasks.

Keywords Multi-modal trajectory · Trajectory representation methods · Deep learning · Contrastive learning

1 Introduction

With the proliferation of GPS-equipped devices and location-based services, enormous trajectories are generated at an unprecedented rate. Such trajectories, capturing rich information about moving objects, play an essential role in various trajectory analysis tasks such as trajectory similarity search, clustering, and prediction [1–5]. Traditional trajectory analysis research requires hand-crafted feature engineering for specific tasks, making it time-consuming and costly. Thus, trajectory representation learning [6–10] has emerged to address this limitation. It automatically captures the characteristics of trajectories and converts raw

✉ Bin Yao
yaobin@cs.sjtu.edu.cn

1 School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, Shanghai, China

2 Cloud Intelligence Group, Alibaba Group, Beijing, China

trajectories into low-dimensional embeddings that can efficiently promote trajectory analysis tasks.

Earlier trajectory representation learning methods leverage seq2seq models to generate representations through reconstruction tasks [7, 10, 11]. However, these methods treat trajectories as ordinary sequence data and fail to fully consider the spatio-temporal information of trajectories. Subsequently, researchers proposed many methods for specific tasks, such as approximate similarity computation [8, 9, 12] and trajectory clustering [13, 14]. However, these methods constrain their applicability to a wide range of downstream tasks. Moreover, with the development of social media, trajectory data containing other modalities, such as text, images, and videos, has emerged. In this paper, we introduce the term *multi-modal trajectory* to represent trajectories that include multiple features such as spatial, temporal, textual, and visual features. Specifically, we focus on multi-modal trajectories incorporating spatial, temporal, and textual modalities due to their wide range of applications. However, existing studies concentrate on spatio-temporal features but ignore additional modal features, potentially leading to poor performance in some scenarios.

Example 1 In the case of trip recommendation, when a tourist plans a trip to a city and queries a route associated with the desired activities, searching for similar travel trajectories will help her make suitable trip plans. In this case, we need to consider spatial, temporal, and textual modalities simultaneously. As shown in Fig. 1, T_1 is a query trajectory, with each location tagged with a specific activity, and T_2 and T_3 are historical travel trajectories. If only spatio-temporal similarity is considered, T_2 would be recommended. However, T_3 shares remarkably similar activities with T_1, represented by the sequence {Museum, Mall, Restaurant}, and also exhibits spatio-temporal similarity. Thus, T_3 is the more suitable trip route to recommend.

Multi-modal features are extensively used in various trajectory analysis and management tasks, including spatial keyword similarity searches on trajectories [15–17], multiple-feature trajectory clustering [18, 19], travel time estimation based on spatio-temporal and POI information [20, 21], and traffic management through cross-retrieval between traffic images and coordinates [22], as detailed in Section 2.3. However, existing trajectory representation studies have not fully utilized these multi-modal features. To bridge this gap, we investigate the multi-modal trajectory representation problem to obtain more expressive, semantically

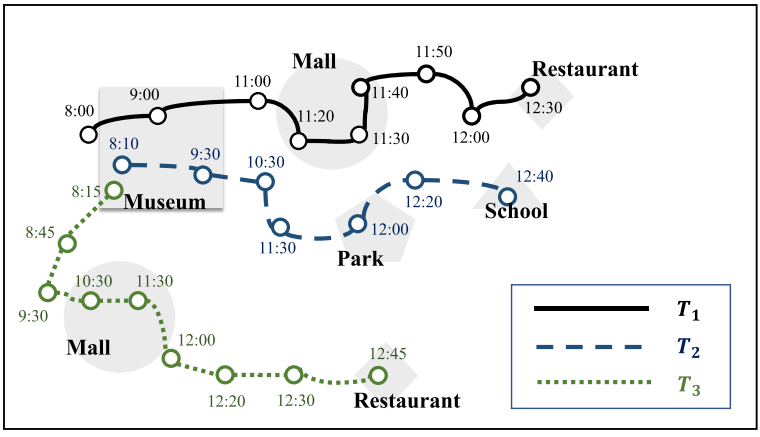


Fig. 1 An example of multi-modal trajectories with spatial, temporal, and textual features

enriched, and generic representations. However, two non-trivial challenges remain to be addressed.

Challenge 1: How to effectively capture and fuse the multi-modal characteristics in trajectories? Existing works [7, 11, 23] employ Word2Vec methods to capture spatial and temporal features but overlook spatial proximity and temporal periodicity. At2vec [11] and At2vec-attn [23] utilize GloVe to capture textual features, yet they may not fully capture the complex contextual relationships within trajectories. Consequently, there is a noticeable absence of methods comprehensively capturing multi-modal characteristics, including spatial proximity, temporal periodicity, and complex textual relationships. Traditional fusion techniques for multi-modal features, which often rely on simple concatenation or weighted combinations, do not adequately consider the correlation and significance of each modality, limiting the potential for generating semantically enriched trajectory representations. Therefore, developing a fusion method that considers the correlations and importance of multi-modal features presents a considerable challenge.

Challenge 2: How to obtain the semantically enriched and generic representations that can be applied to various downstream tasks? The majority of existing studies on trajectory representation have focused on task-specific applications, such as trajectory similarity approximation [8, 9, 24] and trajectory clustering [13, 14]. However, these task-specific approaches tend to generate representations tailored to particular tasks, making it challenging to transfer these representations to other tasks. This limitation narrows their usefulness across a wide array of downstream tasks. Consequently, developing methods to learn intrinsic trajectory features and generate generic trajectory representations applicable to a wide range of downstream tasks remains a significant challenge.

To solve these challenges, we propose a novel and generic **C**ontrastive **L**earning-based **M**ulti-modal **T**rajectory **R**epresentation framework, namely CLMTR. To address Challenge 1, we capture multi-modal characteristics comprehensively by applying a graph embedding method for spatial embeddings to consider spatial proximity, developing a temporal embedding method that considers periodic patterns, and utilizing the pre-trained BERT model for textual embeddings to learn contextual relationships in trajectories. We propose an attention-based fusion method to fuse multi-modal characteristics, adaptively learning different modal weights to derive unified multi-modal embeddings. Furthermore, observing that different modal features within the same trajectory are highly correlated, while those from distinct trajectories exhibit less correlation, we apply a contrastive learning method to compare a trajectory against itself across different modalities. This approach captures the correlations and complementarities among different modal features within the same trajectory, generating fused semantically enriched trajectory representations.

To address Challenge 2, we apply the multi-task learning strategy that jointly trains the intra- and inter-trajectory contrastive learning components. The contrastive learning method can capture the structure and features of trajectories, resulting in more general and discriminative representations. Specifically, the intra-trajectory contrastive learning component focuses on the correlation among different modal features within the same trajectories, while the inter-trajectory contrastive learning component focuses on capturing the relationship among different trajectories. We also propose practical approaches for constructing contrastive views. Through the joint training of intra- and inter-trajectory contrastive learning components, CLMTR can produce expressive and generic representations that are effective across various downstream tasks such as trajectory similarity search, clustering, and travel time estimation.

In summary, our main contributions are as follows.

- We propose multi-modal embedding methods to comprehensively capture multi-modal features, including spatial proximity, temporal periodicity, and complex textual relationships. Furthermore, we propose an attention-based feature fusion method and intra-trajectory contrastive learning method to fuse these multi-modal features adaptively.
- We propose a novel, generic framework CLMTR for multi-modal trajectory representation, which is the first contrastive learning-based solution to the best of our knowledge. CLMTR incorporates intra- and inter-trajectory contrastive learning components, effectively capturing the correlations among multi-modal features within the same trajectories and the relationships among different trajectories, thereby generating semantically enriched and generic representations applicable to a wide range of downstream tasks.
- We conduct extensive experiments on two real-world trajectory datasets. The experimental results demonstrate the superiority of CLMTR over the six state-of-the-art methods across three downstream tasks.

The remainder of the paper is organized as follows. Section 2 discusses the related work. Preliminaries and the problem statement are given in Section 3. Section 4 presents the details of CLMTR. The experimental results are given in Section 5. Finally, Section 6 concludes this paper.

2 Related work

2.1 Trajectory representation learning

Trajectory representation learning aims to transform trajectories into generic low-dimensional vectors suitable for various downstream applications. T2vec [7] utilizes a seq2seq model to reconstruct trajectories from the low-quality ones. Traj2vec [10] extracts pre-defined feature sequences from raw trajectories and trains a seq2seq model using reconstruction loss. AdvTraj2Vec [6] enhances the robustness of t2vec by perturbing the embedding layer parameters. However, these approaches treat trajectories as ordinary sequence data, failing to exploit their spatio-temporal semantic characteristics thoroughly. They also focus on individual trajectory characteristics, neglecting inter-trajectory relationships and failing to capture comprehensive and generic trajectory features. Subsequently, many methods have been proposed for specific tasks, such as approximate similarity computation and trajectory clustering. NeuTraj [8] utilizes an LSTM with a spatial attention memory module to approximate trajectory similarities. Traj2SimVec [24] reduces the training cost of NeuTraj through an index-based sampling strategy and introduces an auxiliary loss for capturing the matching relationships between trajectories. TMN [25] utilizes a matching mechanism to compute attention weights for point pairs across trajectories to improve accuracy. T3S [12] applies the RNN-based model and self-attention to learn trajectory similarities by capturing various unique characteristics of trajectories. GTS [26] learns features from both POIs and trajectories for similarity computation. GTS$^+$ [27] learns spatio-temporal characteristics and spatial network structure to obtain trajectory representations. TrajGAT [9] introduces a graph-based attention model to capture long-term dependencies within trajectories. Trembr [28] exploits the underlying road networks to learn spatio-temporal properties in trajectories. ST2Vec [29] considers fine-grained spatio-temporal relations between tra-

jectories. DETECT [14] and E2DTC [13] focus on trajectory clustering by training with reconstruction and cluster-oriented losses. However, these methods are task-specific, limiting their general applicability to various downstream tasks. They focus on spatio-temporal features, neglecting other modal features. Recently, several methods considering multi-modal features have emerged. At2vec [11] and At2vec-attn [23] learn activity trajectory representations by considering spatio-temporal characteristics and activity semantics jointly. However, they do not capture the correlation of different modal features within the same trajectory and the intricate relationships between different trajectories, thus hindering the generation of semantically enriched trajectory representations.

2.2 Contrastive learning in trajectory

Contrastive learning has recently achieved remarkable success in computer vision [30, 31] and natural language processing [32–34]. This method constructs positive and negative samples, maximizing the similarity between positive pairs while minimizing it between negative pairs. It is frequently employed in extracting features from images and videos. For instance, training the network to identify image orientation and remain invariant to data augmentation can yield high-quality image representations [31]. These representations can subsequently be leveraged for downstream tasks. Recently, a few works based on contrastive learning have emerged in trajectory analysis tasks. CTLTR [35] employs contrastive learning to capture the intrinsic POI dependencies and traveling intent, thereby learning robust representations applicable to tour planning. CL-TSim [36] and CSTRM [37] employ contrastive learning to derive latent trajectory representations. TrajCL [38] introduces a dual-feature, self-attention-based contrastive learning model to extract spatial and structural features from trajectories jointly. START [39] learns trajectory representations by exploiting temporal regularities and travel semantics, introducing trajectory contrastive learning as a self-supervised task. However, these studies do not capture the correlation and complementarity of different modal features within the same trajectory, failing to generate semantically enriched and generic trajectory representations.

2.3 Multi-modal trajectory-based tasks

With the development of location-based services, multi-modal trajectories that include spatial, temporal, textual, and visual features are being rapidly generated. These features are utilized in various trajectory applications. For instance, [15–17] investigate spatial keyword similarity searches on multi-modal trajectories to find trajectories that are not only close geographically but also meet the query's semantic requirements. Zheng et al. [15] address approximate keyword queries by proposing a hybrid index and spatio-textual utility function. Zheng et al. [16] introduce a top-k spatial keyword query, offering a novel similarity function, hybrid indexing structure, and efficient search algorithm. Song et al. [17] present collective spatial keyword queries, integrating spatial, textual, and popularity information with a novel hybrid index. For similarity search and clustering of multi-modal trajectories, [18] proposes a weighted similarity measurement considering all modalities, and [19] introduces a trajectory similarity measurement that considers the semantic relationships between modalities. The travel time estimation task aims to predict the travel time of a route by capturing its inherent features. Li et al. [20], Han et al. [21] combine POI, spatial, and temporal features for more accurate estimation. For traffic management tasks such as traffic accident analysis, [22] presents a cross-modal retrieval method between trajectory images and coordinates. However,

these studies are task-specific and do not fully explore the correlations and complementarities among different features, nor the complex relationships between trajectories.

3 Preliminaries

In this section, we introduce preliminary concepts and formally define the problem of multi-modal trajectory representation learning.

Definition 1 (Multi-modal trajectory) A multi-modal trajectory T is a sequence of time-ordered points containing multi-modal features, e.g., spatial, temporal, textual, imagery, or video features. In this paper, we focus on the spatial, temporal, and textual features of trajectories, and our framework can also be applied to other modal features. $T = \{p_1, p_2, ..., p_{|T|}\}$, where each point $p_i = (p_i.l, p_i.t, p_i.\Phi)$ consists of $p_i.l$ and $p_i.t$, representing the location and the timestamp, respectively. Additionally, $p_i.\Phi = \{w_j\}$ is a set of keywords representing the textual description associated with the points. In the following, we interchange the terms "trajectory" and "multi-modal trajectory" when no ambiguity arises.

Definition 2 (Trajectory encoder)To better handle long dependencies and capture relationships between different points in the trajectory, we use the multi-head self-attention Transformer [40] as the trajectory encoder.

The input to the encoder is fused embeddings of trajectories, denoted as $\mathbf{X} \in \mathbb{R}^{|T| \times d}$. We add positional information into the input using position encoding. For the j-th dimension of the fused embedding of the i-th point on a trajectory, represented by $\mathbf{X}_{i,j}$, we update it by adding the corresponding position encoding value $e_{i,j}$ as follows:

$$\mathbf{X}_{i,j} = \mathbf{X}_{i,j} + e_{i,j} \tag{1}$$

$$e_{i,j} = \begin{cases} \sin(i/10000^{2j/d}), & 0 \leq j < d \text{ and j is even} \\ \cos(i/10000^{2j/d}), & 0 \leq j < d \text{ and j is odd} \end{cases} \tag{2}$$

where d is the embedding dimension.

In the multi-head self-attention Transformer, the $\mathbf{H}_2$ attention heads transform $\mathbf{X}$ into the $\mathbf{H}_2$ query matrix $\mathbf{Q}_h = \mathbf{X}\mathbf{W}_h^Q$, the key matrix $\mathbf{K}_h = \mathbf{X}\mathbf{W}_h^K$, and the value matrix $\mathbf{V}_h = \mathbf{X}\mathbf{W}_h^V$, where $\mathbf{W}_h^Q, \mathbf{W}_h^K, \mathbf{W}_h^V \in \mathbb{R}^{d \times d'}$ are learnable parameters, and $d' = d/\mathbf{H}_2$. The h-th attention head is computed as (3), and the number of attention heads is set to 4 in the experiment.

$$A_h(\mathbf{Q}_h, \mathbf{K}_h, \mathbf{V}_h) = \text{softmax}\left(\frac{\mathbf{Q}_h \mathbf{K}_h^T}{\sqrt{d'}}\right)\mathbf{V}_h \tag{3}$$

The outputs from attention heads are concatenated and then passed through a projection matrix $\mathbf{W}^O \in \mathbb{R}^{d \times d}$, resulting in the outputs $\mathbf{X}' \in \mathbb{R}^{|T| \times d}$ as follows:

$$\mathbf{X}' = \text{MultiAtt}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = (\text{A}_1 || \cdots || \text{A}_{\text{H}_2})\mathbf{W}^{\text{O}} \tag{4}$$

Then we incorporate layer normalization and residual connection to enhance the model's performance. Ultimately, we utilize a feed-forward network (FFN) comprising two linear layers and a ReLU activation function to obtain the output representation $\mathbf{v}$ of trajectory T as follows:

$$\mathbf{v} = (\text{ReLU}(\mathbf{X}'\mathbf{W}_F^1 + b_F^1))\mathbf{W}_F^2 + b_F^2 \tag{5}$$

where $\mathbf{W}_F^1, \mathbf{W}_F^2 \in \mathbb{R}^{d\times d}$, $b_F^1, b_F^2 \in \mathbb{R}^d$ are learnable parameters. The layer normalization and residual connection are also applied here.

Problem 1 Given a multi-modal trajectory dataset $D = \{T_i\}_{i=1}^{|D|}$, our objective is to propose a self-supervised framework for training a trajectory encoder $E : T_i \rightarrow \mathbf{v}_i$ that maps trajectory T_i to a generic d-dimensional representation vector $\mathbf{v}_i \in \mathbb{R}^d$, effectively capturing the spatial-temporal characteristics and textual information of the trajectory. The representations can be applied to various downstream tasks, such as trajectory similarity search, trajectory clustering, and travel time estimation.

4 Methodology

In this section, we introduce the CLMTR framework as illustrated in Fig. 2. We begin by explaining the multi-modal features embedding component, followed by detailing the intra- and inter-trajectory contrastive learning components.

4.1 Multi-modal features embedding

Location embedding Spatial location plays a crucial role in capturing the shape of a trajectory and the connectivity among its points. Traditional methods divide space into a grid, assigning points to corresponding grid cells and transforming trajectories into sequences of grid cell IDs. Spatial embeddings are derived using the Word2Vec method. However, these methods neglect the connectivity and spatial proximity among points. To address this limitation, we adopt the Node2Vec technique. Instead of considering all grid cells, we focus only on those called prominent cells, to which a higher number of trajectory points are assigned, mitigating the issue of sparse information. We construct a graph in which each vertex corresponds to a prominent grid cell, and edges connect each vertex to its k nearest neighbors.

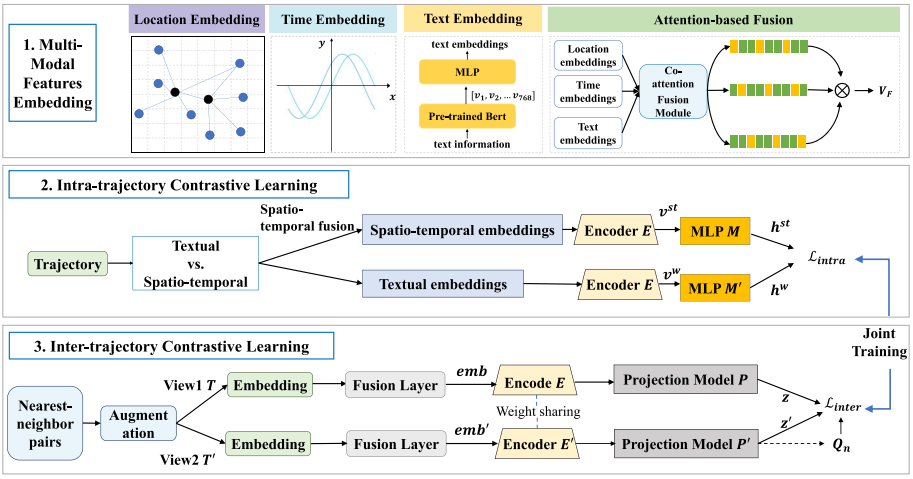


Fig. 2 The framework of CLMTR consists of three main components: the top part is the multi-modal features embedding component, the middle part is the intra-trajectory contrastive learning component, and the bottom part is the inter-trajectory contrastive learning component

Using the Node2Vec algorithm, we approximate the spatial conditional probability of vertices within their respective neighborhoods. As a result, locations sharing similar neighborhoods exhibit similar embeddings. These vertex embeddings serve as spatial embeddings, allowing the location embedding of a trajectory to be represented as $T_s = \langle l_1, l_2, \cdots, l_n \rangle$, where l_i corresponds to the location embedding of the trajectory point p_i.

Time embedding One straightforward approach is to divide the time axis into discrete slots, assigning trajectory points to these slots based on their timestamps. The Word2Vec method can then be used to generate temporal embeddings by utilizing slot IDs to represent temporal trajectory points. However, this method does not effectively capture the periodic patterns of time. Temporal features exhibit strong periodicity, manifesting in seconds, minutes, hours, days, etc. Therefore, the time embedding must be scale-independent and capable of capturing the periodic nature of time. We employ a sine function to model temporal features more accurately. By learning parameters from time inputs, the model captures the periodic characteristics of time. This enhances the understanding and utilization of temporal information, making it less affected by uneven sampling rates. Specifically, let $t_{i,j}$ represent the j-th dimensional value of the time embedding for the i-th point on a trajectory.

$$t_{i,j} = \begin{cases} \omega_j t + \varphi_j, & \text{if } j = 0 \\ \sin(\omega_j t + \varphi_j), & \text{if } 1 \leq j \leq d_{im} - 1 \end{cases} \tag{6}$$

Here, ω_j and φ_j are learnable parameters, d_{im} denotes the embedding dimension, $\sin(\cdot, \cdot)$ is a periodic activation function that captures periodic behaviors, and the linear term within the equation captures non-periodic patterns. Therefore, the time embedding of a trajectory can be denoted as $T_t = \langle t_1, t_2, \cdots, t_n \rangle$, where t_i represents the time embedding of the trajectory point p_i.

Text embedding The textual information associated with trajectory points, indicating user activities, check-in tips, or POI information, provides enriched semantic information for trajectories. We use the pre-trained BERT model for text feature extraction to capture the complex contextual relationship within the trajectory, generating embeddings for text associated with each trajectory point. The tokenizer in BERT utilizes the WordPiece technique, which is a subword-based approach for breaking down each word into a sequence of subword units. The pre-trained BERT model used in our experiment is *bert-base-uncased*. The text embeddings derived from BERT have a size of 768 dimensions. Subsequently, a linear model is used to reduce the dimensionality of these text embeddings from 768 to 256. Eventually, the text embedding of a trajectory is denoted as $T_w = \langle w_1, w_2, \cdots, w_n \rangle$, where w_i corresponds to the text embedding of the trajectory point p_i.

Attention-based fusion Following the multi-modal feature embedding process, we acquire the location embedding $T_s = \langle l_1, l_2, \cdots, l_n \rangle$, time embedding $T_t = \langle t_1, t_2, \cdots, t_n \rangle$, and text embedding $T_w = \langle w_1, w_2, \cdots, w_n \rangle$ of a trajectory, respectively. Given that these representations capture distinct aspects of trajectories, we enhance their expressive power by facilitating their interaction. Thus, we employ an attention mechanism to calculate the attention scores associated with the representation embedding and subsequently perform a weighted fusion, effectively leveraging the importance of different modalities. Firstly, we transform spatial, temporal, and textual embeddings via a matrix $\mathbf{W}_v$.

$$\tau_1 = \mathbf{W}_v T_s, \quad \tau_2 = \mathbf{W}_v T_t, \quad \tau_3 = \mathbf{W}_v T_w \tag{7}$$

Then the interaction of these multi-modal features is computed as follows:

$$\begin{aligned} \beta_{i,j} &= \frac{\exp\left(\mathbf{W}_Q \tau_i \cdot \mathbf{W}_K {\tau_j}^T\right)}{\sum_{j' \in \{1,2,3\}} \exp\left(\mathbf{W}_Q \tau_i \cdot \mathbf{W}_K {\tau_{j'}}^T\right)}, \\ V_s &= \text{Norm}\left(\text{FFN}\left(\beta_{1,1}\tau_1 + \beta_{1,2}\tau_2 + \beta_{1,3}\tau_3\right) + T_s\right), \\ V_t &= \text{Norm}\left(\text{FFN}\left(\beta_{2,1}\tau_1 + \beta_{2,2}\tau_2 + \beta_{2,3}\tau_3\right) + T_t\right), \\ V_w &= \text{Norm}\left(\text{FFN}\left(\beta_{3,1}\tau_1 + \beta_{3,2}\tau_2 + \beta_{3,3}\tau_3\right) + T_w\right), \end{aligned} \tag{8}$$

$$V_F = \text{Concat}(V_s, V_t, V_w) \tag{9}$$

where $\mathbf{W}_Q$ and $\mathbf{W}_K$ are matrices of the same shape as $\mathbf{W}_V$, V_s, V_t, and V_w are the enhanced embeddings of T_s, T_t, and T_w, respectively. The fused multi-modal embedding is obtained through (9).

4.2 Intra-trajectory contrastive learning component

We observe a strong correlation among various modal features within the same trajectory, contrasted with the reduced correlation between modalities from different trajectories. To enhance the collaboration between different modalities and capture the inherent characteristics of trajectories, we introduce an intra-trajectory contrastive learning component that compares a trajectory with itself across a range of modalities. Through this component, learning in one modality can influence learning in others, ensuring that only high-quality representations across different modalities are aligned. It is not trivial to construct suitable contrastive views for multi-modal features. We present the solution based on the following two observations. First, we note that single-modal features can complement each other. For instance, knowing a location allows us to infer the activities taking place there. Moreover, the more modal features we have, the more accurate our inferences become. Specifically, specifying both the location and the corresponding timestamp enables more precise predictions about the activities occurring at that time and place. Consequently, we propose two strategies for creating contrastive views: contrasting single modal features and contrasting a single feature with multiple fused features. We can create six contrastive views in total: spatial vs. temporal feature, spatial vs. textual feature, temporal vs. textual feature, spatial feature vs. temporal-textual fused features, temporal feature vs. spatial-textual fused features, and textual feature vs. spatio-temporal fused features. Preliminary experimental results indicate that selecting textual vs. spatio-temporal fused features as the contrastive view is particularly effective. Therefore, we adopt this view as our default contrastive view. Contrastive views from the same trajectory serve as positive samples, while those from different trajectories are negative. Our approach is generic and can be adapted for trajectories with a variety of modal features, fully leveraging the complementarity and correlations between different modalities.

Figure 2 shows an example of generating a contrastive view of the textual vs. spatio-temporal fused features. For trajectory T, we obtain the location, time, and text embeddings and generate a fused spatio-temporal embedding as described in Section 4.1. The spatio-temporal and textual embeddings are then input into a trajectory encoder, producing spatio-temporal and textual vectors $\mathbf{v}^{st}$ and $\mathbf{v}^{w}$, respectively. Subsequently, two one-layer MLPs, M and M', process $\mathbf{v}^{st}$ and $\mathbf{v}^{w}$ to generate lower-dimensional representations $\mathbf{h}^{st}$ and $\mathbf{h}^{w}$ as follows:

$$\mathbf{h}^{st} = \mathbf{v}^{st}\mathbf{W}_M + b_M, \quad \mathbf{h}^{w} = v^{w}\mathbf{W}_{M'} + b_{M'} \tag{10}$$

where $\mathbf{W}_M$ and b_M are learnable parameters of M, $\mathbf{W}_{M'}$ and $b_{M'}$ are learnable parameters of M', and $\mathbf{h}^{st}$ and $\mathbf{h}^{w}$ form positive pairs for contrastive learning.

We employ the normalized temperature-scaled cross-entropy loss [31] as the contrastive objective. The loss function is defined as:

$$\mathcal{L}_{intra} = -\frac{1}{N}\sum_{i=1}^{N}\log\frac{\exp(d(\mathbf{h}_i, \mathbf{h}_i{}')/\tau)}{\sum_{j=1}^{N}\mathbf{1}_{[i\neq j]}\exp(d(\mathbf{h}_i, \mathbf{h}_j)/\tau)} \tag{11}$$

Here, $\mathbf{h}_i$ is the anchor sample, while $\mathbf{h}_i{}'$ and $\mathbf{h}_j$ are the positive and negative samples of $\mathbf{h}_i$, respectively. The temperature parameter τ controls the smoothness of the probability distribution over the similarities of samples, influencing the contribution of positive and negative samples to the loss function. $d(\cdot,\cdot)$ is cosine similarity, and $\mathbf{1}$ is an indicator variable that equals one if the condition is satisfied and zero otherwise.

4.3 Inter-trajectory contrastive learning component

To effectively capture relationships among multi-modal trajectories and generate generic representations, we introduce an inter-trajectory contrastive learning component to compare a trajectory with other trajectories. Constructing positive and negative samples for contrastive learning is crucial, as it significantly affects model performance. Existing methods typically construct two variants of the same trajectory and treat these variants as positive samples. For example, for a trajectory T_i, two augmented variants, T_i^1 and T_i^2, are created as positive samples, while variants derived from T_i and another trajectory T_j are considered negative samples. However, if T_i and T_j are similar trajectories, classifying their variants as negative samples may hinder the model's discriminative ability. To address this limitation, we propose a similarity-based strategy to obtain positive and negative samples, using nearest-neighbor trajectories as positive pairs. The distance function $d(T_i, T_j)$ is defined as:

$$d(T_i, T_j) = d_S(T_i^s, T_j^s) + d_T(T_i^t, T_j^t) + d_W(T_i^w, T_j^w) \tag{12}$$

Here, T_i^s, T_i^t, and T_i^w represent the spatial, temporal, and textual sequences of T_i, respectively, and d_S, d_T, and d_W denote the corresponding distances. We can utilize any existing distance measure for calculation. In this paper, we select the Fréchet distance for computing d_S and d_T, and the Edit distance for computing d_W, as they are widely recognized distance measurements. We use the distance between trajectories to measure their similarity: the smaller the trajectory distance, the higher the similarity between the trajectories. To enhance efficiency, we use a k-d tree to index trajectories and retrieve the k-nearest neighbors for each trajectory via a k-d tree-based query algorithm. The nearest neighbor becomes a positive sample, while trajectories beyond the k-nearest neighbors are considered negative samples.

Furthermore, we propose four data augmentation strategies to help the model capture distinguishing features from different variants and enhance robustness: (1) **Downsampling**: Downsampling reduces the sample rate of trajectories, which can increase robustness and mitigate overfitting. Specifically, we randomly discard points from the trajectory T to generate a variant T'. The start and end points are preserved to maintain the route integrity. For the remaining points, we randomly determine the probability p of whether to drop each point. If p is less than 0.4, the point is discarded.

(2) **Distorting**: Distorting introduces noise to trajectories, further enhancing the model's robustness. Specifically, we randomly add an offset to each point in trajectory T. For spatial distortion, we add a minor location offset to each point p_i, and the distorted location

coordinate is defined as:

$$p_i'.l = (x_i + \Delta x_i, y_i + \Delta y_i), \Delta x_i \sim X_n, \Delta y_i \sim X_n, X_n \sim d_m \mathcal{N}(\mu, \sigma^2) \tag{13}$$

where X_n is a bounded Gaussian distribution, and d_m denotes the maximum location offset, which has a default value of 50 meters. We set the default values of μ and σ to 0 and 0.5, respectively. Similarly, for temporal distortion, we add random noise to the timestamp of each point as follows:

$$p_i'.t = t_i + \Delta t_i, \Delta t_i \sim X_n, X_n \sim t_m \mathcal{N}(\mu, \sigma^2) \tag{14}$$

where t_m represents the maximum time offset with a default value of 200 seconds. The default values of μ and σ are 0 and 0.5, respectively. For textual distortion, if a POI contains only one word, we omit it and proceed to the next. Otherwise, for each word in POI, we randomly determine the probability p for the operation as follows: if $p < 0.05$, we drop the word; if $0.05 \leq p < 0.1$, we insert a word after it; if $0.1 \leq p < 0.15$, we replace the word. The word to insert or replace is randomly selected from the existing keyword set.

(3) **Trimming**: Trimming creates sub-trajectories by cutting a prefix or a suffix (or both) from trajectories, enabling the model to consider local patterns and detect partially overlapped trajectories. Specifically, we remove a prefix, suffix, or both from trajectory T, creating the variant T' with the remaining points. To control the proportion of points retained in T', we introduce a parameter β with a default value of 0.8. The T' is represented as:

$$T' = \{p_i, p_{i+1}, \cdots, p_{\lfloor i+\beta|T| \rfloor}\}, \quad i \in \{1, 2, \cdots, \lceil (1-\beta)|T| \rceil\} \tag{15}$$

(4) **Simplification**: Simplification identifies critical trajectory points and removes less critical ones, aiding the model in capturing distinctive features more effectively. We use the Douglas-Peucker (DP) algorithm [41], a broadly applicable method, to simplify trajectories. The DP algorithm starts by drawing a line segment between the endpoints of trajectory T. It identifies the point furthest from this line segment, creating two new segments connecting this point to each endpoint. This process repeats recursively until all points are sufficiently close to the line segments, with a threshold τ_d of 50 meters.

Note that the choice of parameters for data augmentation is based on our empirical and experimental findings.

The augmented variants of positive pairs, i.e., nearest-neighbor pairs, remain positive samples, while the variants of negative pairs continue to be negative samples. This approach facilitates more general and beneficial semantic transformations for downstream tasks. After acquiring positive and negative samples, we proceed with training as depicted in Fig. 2. Positive samples T and T' are fed into the Multi-Modal Features Embedding module to achieve fused embeddings. These embeddings are processed by trajectory encoders E and E' to produce representation vectors, which are then mapped to lower-dimensional vectors $\mathbf{z}$ and $\mathbf{z}'$ by two projection models P and P', each comprising two fully connected layers with ReLU activation. Negative samples are processed using the same procedure, producing vectors denoted as $\mathbf{z}^-$. A fixed-size queue, Q_n, stores negative samples from current and recent past batches. We apply the InfoNCE [42] loss for model training as follows:

$$\mathcal{L}_{inter} = -\log \frac{\exp(d(\mathbf{z}, \mathbf{z}')/\tau)}{\exp(d(\mathbf{z}, \mathbf{z}')/\tau) + \sum_{j=1}^{|Q_n|} \exp((\mathbf{z}, \mathbf{z}_j^-)/\tau)} \tag{16}$$

During training, we reuse negative samples from Q_n and adopt the momentum update method to update the parameters for E' and P' smoothly.

$$\theta_{E'} = \alpha\theta_{E'} + (1-\alpha)\theta_E, \quad \theta_{P'} = \alpha\theta_{P'} + (1-\alpha)\theta_P \tag{17}$$

where θ_E and $\theta_{E'}$ are the parameters of encoders E and E', and θ_P and $\theta_{P'}$ are the parameters of models P and P', respectively. α is a momentum coefficient that controls the parameter updates, with a default value of 0.999.

4.4 Multi-task learning

Finally, we incorporate the intra- and inter-trajectory contrastive learning components into a multi-task learning framework, with the overall training loss function aggregating the intra- and inter-trajectory contrastive losses as follows:

$$\mathcal{L}_{overall} = \mathcal{L}_{intra} + \mathcal{L}_{inter} \tag{18}$$

L_{intra} and L_{inter} are of the same magnitude, so we set their weights equally to capture both intra- and inter-trajectory features simultaneously. The weights can be adjusted to prioritize either intra- or inter-trajectory features for different applications, allowing the model to focus on the desired characteristics accordingly. After model training, we can apply the encoder E to generate trajectory representations. By jointly leveraging the intra- and inter-trajectory contrastive learning components, CLMTR can generate semantically enriched and generic representations applicable to diverse downstream tasks.

5 Experiments

In this section, we conduct extensive experiments to evaluate the performance of CLMTR. Firstly, we compare the performance of our method with the state-of-the-art methods on two real-world datasets for three downstream tasks. Secondly, we adopt ablation studies to verify the effectiveness of critical modules of CLMTR. Thirdly, we study the impact of model parameters. At last, we investigate the efficiency and scalability of CLMTR.

5.1 Experimental setup

Datasets We use two real-world trajectory datasets: Geolife [43] and T-Drive [44]. We employ the AMAP API to enrich these datasets with Points of Interest (POIs) for each trajectory point. The enriched versions of these datasets are referred to as Geolife$^+$ and T-Drive$^+$, respectively. Following state-of-the-art methods like CL-Tsim [36] and TrajCL [38], and considering the training time and storage requirements, we set the maximum trajectory length to 200 points. Our goal is to propose a solution for multi-modal trajectory representation rather than managing long trajectories. Future research will focus on capturing the multi-modal features in long trajectories efficiently and effectively. Thus, trajectories exceeding 200 points are segmented into smaller sub-trajectories, while those with fewer than 20 points are excluded from datasets. The details of datasets are summarized in Table 1. We shuffle each dataset and divide it into three disjoint subsets: 70% for training, 10% for validation, and 20% for testing.

Evaluation metrics (1) For the trajectory similarity search task, we use Mean Rank (MR) to evaluate the most similarity search and use HR@5 and R5@20 to evaluate the kNN search. Here, HR@k signifies the hit ratio between the top-k results and the corresponding ground-truth results, and Rk@t represents the recall of the top-k ground truth within the top-t

Table 1 Dataset statistics

Description	Geolife$^+$	T-Drive$^+$
Number of trajectories	107,843	258,704
Number of POIs	4,474	4,531
Number of points	19,412,740	17,740,902
Number of keywords per POI	5.78	4.52
Number of keywords	4,483	4,536

results. (2) For the trajectory clustering task, we use Accuracy (ACC), Rand Index (RI), and Normalized Mutual Information (NMI) to evaluate the performance. (3) For the travel time estimation task, we report Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE), and Root Mean Square Error (RMSE).

Baselines We compare CLMTR with six state-of-the-art approaches:

- **At2vec** [11] employs the skip-gram model to obtain spatial and temporal embeddings and utilizes the GloVe model to obtain textual embeddings. These embeddings are then concatenated to form the fusion embedding. Next, At2vec leverages a seq2seq model to learn trajectory representations.
- **At2vec-attn** [23] modifies the multi-modal feature fusion method in At2vec by introducing a multi-layer attention mechanism.
- **E2DTC** [13] focuses on learning cluster-friendly trajectory representations through joint training with reconstruction loss and cluster-oriented loss.
- **ST2Vec** [29] is a supervised method to learn trajectory representations, which employs spatio-temporal trajectory similarities as training signals to approximate the similarity function.
- **CL-TSim** [36] utilizes a contrastive learning mechanism to learn trajectory representations for effective and efficient similarity computation.
- **TrajCL** [38] proposes a dual-feature self-attention-based contrastive learning framework for obtaining trajectory representations.

For methods that focus on spatial and temporal features, including E2DTC, ST2Vec, CL-TSim, and TrajCL, we have extended them by integrating the multi-modal feature embedding and fusion methods used by At2vec, thus supporting multi-modal trajectory representation learning.

Parameter settings We set the fused embedding dimension to 256. The number of encoder layers is 2, and the number of attention heads is 4. The length of the grid cells is set to 50 meters. The maximum number of training epochs is 50, and we early stop after 10 consecutive epochs without improvements in the loss. We use the Adam optimizer with an initial learning rate of 0.001, which is decayed by 0.5 every 10 epochs. Moreover, the batch size is set to 256, the negative samples queue size is 2,048, and the temperature parameter is 0.05.

Environment settings Experiments are conducted on a Linux server with CUDA 11.7 and two NVIDIA GeForce RTX 3090 GPUs, each with 24GB memory. All approaches are implemented in Python 3.9 with PyTorch 2.0.1.

5.2 Performance comparison

Overall performance Table 2 presents the overall results. We use '↑' (and '↓') to indicate that a larger (and smaller) result is better. The best results are highlighted in **bold**. The Euclidean

Table 2 Three downstream tasks overall performance on T-Drive^{+} and Geolife^{+}

	Methods	Similarity search			Trajectory clustering			Travel time estimation		
		MR ↓	HR @5 ↑	R10 @50 ↑	ACC ↑	NMI ↑	RI ↑	MAE ↓	MAPE ↓	RMSE ↓
T-Drive^{+}	At2vec	26.25	0.48	0.89	0.52	0.71	0.77	2073.56	34.31	2484.23
	At2vec-attn	10.80	0.27	0.68	0.55	0.68	0.83	2153.92	49.21	2573.49
	E2DTC	25.93	0.39	0.78	**0.89**	**0.76**	0.81	1632.29	30.23	2275.73
	ST2Vec	31.41	0.47	0.86	0.30	0.63	0.70	2727.96	34.57	2788.16
	CL-TSim	15.71	0.46	0.85	0.47	0.74	0.81	1210.88	18.92	1593.15
	TrajCL	3.50	0.47	0.96	0.81	0.46	0.58	1434.59	19.45	1360.23
	CLMTR	**1.03**	**0.50**	**0.98**	0.87	0.74	**0.90**	**1122.65**	**13.71**	**1271.66**
Geolife^{+}	At2vec	14.81	0.53	0.88	0.29	0.68	0.66	92.71	18.88	142.75
	At2vec-attn	12.35	0.49	0.81	0.49	0.72	0.78	62.08	14.67	93.73
	E2DTC	14.69	0.52	**0.91**	0.92	0.69	0.77	42.20	13.93	55.35
	ST2Vec	27.95	0.48	0.83	0.84	0.64	0.71	54.17	14.12	84.32
	CL-TSim	7.58	0.51	0.88	0.80	0.62	0.74	63.27	17.96	74.73
	TrajCL	1.08	0.56	0.83	0.60	0.69	0.76	63.25	17.96	74.71
	CLMTR	**1.05**	**0.58**	0.90	**0.97**	**0.74**	**0.79**	**35.36**	**12.72**	**48.15**

The best results of the three tasks are highlighted in bold font

distance between trajectory representations is employed to approximate the trajectory similarity. We can observe that CLMTR outperforms other methods on two real-world datasets for three tasks in almost all metrics. CLMTR surpasses At2vec and At2vec-attn as they overlook the inter-trajectory relationships and correlations among diverse modal features within a trajectory. Although E2DTC performs well on the clustering task as it uses reconstruction and cluster-oriented loss for joint training, its performance in other tasks is comparable to that of At2vec. CLMTR outperforms ST2Vec, which does not comprehensively capture the relationship among different modal features. Additionally, CLMTR surpasses CL-TSim and TrajCL as they fail to consider the correlations among different modal features within same trajectories and lack an effective multi-modal feature fusion method.

Performance of trajectory similarity search (1) Most similar trajectory search. This task aims to find the trajectory $T_a{}'$ in dataset D_d that is most similar to a given query trajectory T_a from dataset D_q. Specifically, we randomly select N_q trajectories from the test dataset, denoted as D_q. For each trajectory T_a in D_q, we generate its downsampled version $T_a{}'$ and include it in the dataset D_d. We then select other N_d trajectories from the test dataset that do not overlap with D_q and add them to D_d. The default values for N_q and N_d are 1,000 and 3,000, respectively. For each T_a in D_q, we compute the similarity between T_a and all trajectories in D_d and determine the MR of $T_a{}'$ by ranking the similarities between T_a and each trajectory in D_d in descending order. Ideally, $T_a{}'$ should rank first, as it is generated from T_a.

(2) k-nearest trajectory search Given a query trajectory, this task aims to find its top-k similar trajectories from the target dataset. For each T_a in D_q, we obtain the kNNs from the dataset D_d as its ground truth. Next, we construct the transformed dataset $D_q{}'$ from D_q through a downsampling approach. Finally, for each query trajectory $T_a{}'$ in $D_q{}'$, we find its kNNs from the dataset D_d and compare the result with the corresponding ground truth. Table 2 shows that CLMTR outperforms all baselines in the most similar and k-nearest search tasks, demonstrating CLMTR's effectiveness in identifying similar trajectories.

Performance of trajectory clustering We randomly select 1,000 trajectories from the test dataset, termed D_c. For each trajectory T_i in D_c, we create two sub-trajectories: one consisting of the odd-indexed points, denoted as $T_i^a = \{p_1, p_3, p_5, \cdots\}$, and the other consisting of the even-indexed points, denoted as $T_i^b = \{p_2, p_4, p_6, \cdots\}$. T_i^a is added to the dataset D_a, and T_i^b is added to the dataset D_b. Subsequently, hierarchical clustering is applied to D_a to derive the ground truth clustering result, and the clustering result of D_b is obtained by the same method. Ideally, the clustering results of D_a and D_b should be identical, as each T_i^a in D_a and each T_i^b in D_b originate from the identical trajectory T_i. As shown in Table 2, CLMTR achieves comparable performance to E2DTC on the T-Drive$^+$ dataset and surpasses other baselines across two datasets because E2DTC is specifically trained for the clustering task. The results indicate that CLMTR effectively generates semantically enriched trajectory representations, leading to accurate performance.

Performance of trajectory travel time estimation This task estimates the travel time from a given origin and destination, considering the trajectory representation and departure time. We employ a two-layer, fully connected network to predict the travel time. The MSE is utilized as the optimization objective for this regression model. Table 2 demonstrates that CLMTR consistently outperforms all baselines because it can capture underlying spatial-temporal semantics, thereby obtaining accurate results.

5.3 Ablation study

We conduct ablation experiments to evaluate the effectiveness of each component in CLMTR. We only present the results on T-Drive$^+$ for brevity as the performance trends observed on Geolife$^+$ are similar. Figure 3 shows the results.

Impact of multi-modal features embedding We evaluate this component with the following variants: (1) *w/ L-SkipGram*: this variant utilizes the skip-gram algorithm for location embedding. (2) *w/ T-SkipGram*: the variant employs a skip-gram model for time embedding. (3) *w/ Glove*: the variant adopts the Glove model for text embedding. (4) *w/ Concat*: this variant employs the concatenation method to obtain the multi-modal fused embedding. As shown in Fig. 3, CLMTR outperforms *w/ L-SkipGram*, demonstrating the effectiveness of our location embedding technique in capturing spatial proximity. Similarly, CLMTR outperforms *w/ T-SkipGram* because *w/ T-SkipGram* ignores periodic temporal patterns. Furthermore, CLMTR is superior to *w/ Glove*, as the latter fails to capture complex contextual relationships effectively. CLMTR surpasses *w/ Concat*, indicating the limitations of the concatenation method in adaptively leveraging the weights of different modal features within a trajectory.

Impact of intra-trajectory contrastive learning component We evaluate this component with the variant, denoted as *w/o Intra-CL*, which lacks the intra-trajectory contrastive learning method. The result shows that CLMTR outperforms *w/o Intra-CL* because the latter does not effectively capture the correlations and complementarities among different modal features within the same trajectory, failing to generate semantically enriched representations.

Impact of inter-trajectory contrastive learning component We evaluate this component with several variants: (1) *w/o Inter-CL*: this variant removes the inter-trajectory contrastive learning method from CLMTR. (2) *w/ LSTM*: the variant substitutes the trajectory encoder with an LSTM. The result demonstrates that CLMTR is superior to *w/o Inter-CL*, thereby indicating the effectiveness of the inter-trajectory contrastive learning in capturing trajectory relationships and enhancing trajectory-level feature learning. Furthermore, *w/ LSTM* yields the worst performance among all variants due to the inherent limitations of LSTMs in capturing long-term dependencies within trajectories.

Impact of data augmentation strategies Data augmentation strategies are critical in contrastive learning for capturing multi-modal characteristics and improving model performance. We show performance with different method pairs to explore the effectiveness of our four proposed trajectory data augmentation strategies. For brevity, we present only the similarity

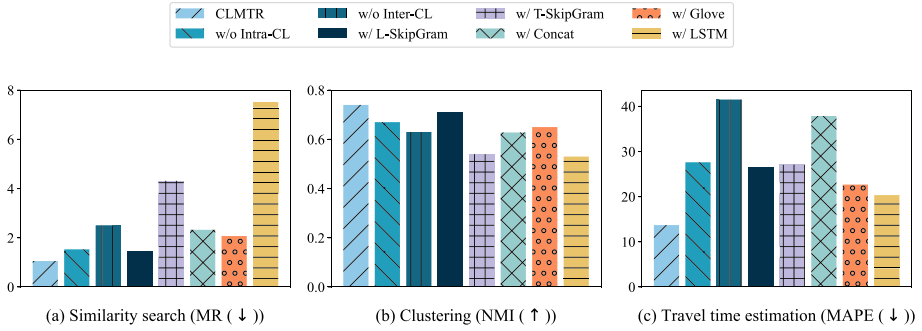
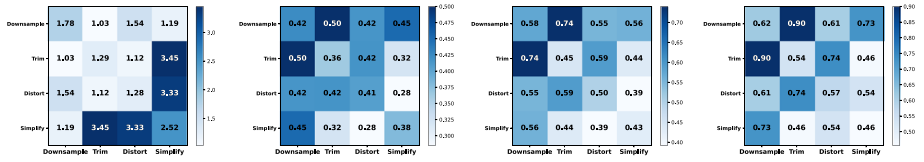


Fig. 3 Ablation study results on T-Drive$^+$



(a) MR(↓) (b) HR@5 (↑) (c) NMI (↑) (d) RI (↑)

Fig. 4 Impact of data augmentation methods

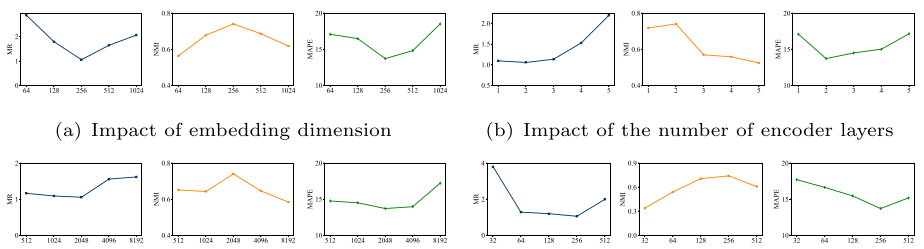
search and clustering results on the T-Drive$^+$ dataset. Figure 4 shows the performance of different combinations of augmentation strategies. In Fig. 4(a), lighter colors represent better performance, while darker colors indicate better performance in the remaining subfigures. The results confirm the importance of the proposed augmentation methods. For example, we observe that downsampling is a simple yet effective strategy. Overall, downsampling and trimming perform best in both tasks. Consequently, we adopt them as our default augmentation methods.

5.4 Parameter study

We further conduct parameter sensitivity analysis for crucial hyperparameters. Note that a smaller MR and MPAE value indicates better performance, whereas a larger NMI value corresponds to better results. As shown in Fig. 5(a), the performance initially improves as the embedding dimension d increases, followed by a decline when d becomes excessively large since a higher embedding dimension may lead to overfitting. In Fig. 5(b), the performance improves initially as the number of encoder layers l increases but then drops when l exceeds 2, as more layers may also lead to overfitting. As depicted in Fig. 5(c), the performance declines when the negative sample queue N_q is too large since a large queue may include too many "hard" negative samples that differ marginally from the given anchor. Figure 5(d) shows that the performance improves as the batch size increases but drops when the batch size reaches 512, as a large batch may also include too many "hard" negative samples. We select the hyperparameter values that yield the best results as the default values for the experiments.

5.5 Model efficiency and scalability

As shown in Table 3, the training time of CLMTR is about one hour and ten minutes, which is acceptable in practice. The complex recurrent operations of At2vec, At2vec-attn, and E2DTC



(a) Impact of embedding dimension (b) Impact of the number of encoder layers

(c) Impact of negative samples queue size (d) Impact of training batch size

Fig. 5 Parameter sensitivity analysis on T-Drive$^+$

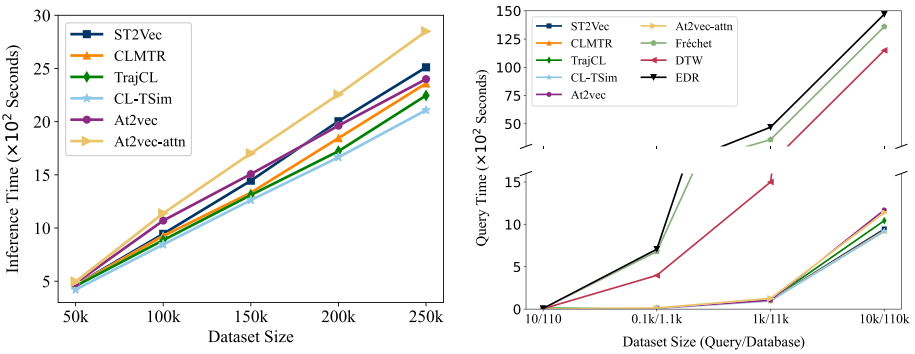
Table 3 Training time (seconds) on T-Drive$^+$ and Geolife$^+$

	CL-TSim	CLMTR	TrajCL	ST2Vec	At2vec-attn	At2vec	E2DTC
T-Drive	4,108	4,210	5,835	6,950	6,980	7,490	9,500
Geolife	3,960	4,015	5,670	6,524	6,670	7,260	9,324

contribute to their longer training times. Figure 6(a) shows the inference time of embedding 50k-250k trajectories. Since At2vec and E2DTC share the same encoder structure, we omit the result of E2DTC. CLMTR is more efficient than methods based on RNN models, such as At2vec, At2vec-attn, and ST2Vec. This is because RNN models require $O(L)$ sequential operations to process a trajectory, whereas a self-attention model only requires an $O(1)$ operation. Here, L represents the length of the trajectory. CLMTR is slightly slower than other self-attention models as it involves the intra-trajectory contrastive learning component. Figure 6(b) shows time costs for the most similar trajectory search. The time costs of learning-based approaches encompass the time for representation generation and similarity computation. We observe that learning-based approaches exhibit a substantial speed advantage, exceeding traditional trajectory methods by an order of magnitude. Moreover, both the inference time and the time required for similarity search of CLMTR increase linearly with the data volume, indicating that CLMTR is scalable for large datasets.

5.6 Discussion

In this section, we analyze the limitations of CLMTR and propose future research directions. First, due to constraints related to training time and storage requirements, we set the maximum trajectory length to 200 points. In the future, we aim to develop methods that can effectively capture the intricate long-term dependencies of multi-modal features in longer trajectories. We also plan to explore lightweight models to further reduce computational costs and improve scalability. Additionally, we currently use pre-trained BERT to extract textual features from trajectories. In the future, we intend to fine-tune BERT specifically for trajectory-related tasks, thereby adapting it more effectively to the spatio-temporal domain. Lastly, while we currently focus on trajectories in Euclidean space, many traffic-related



(a) Representation generation time (b) Similarity search time

Fig. 6 Model efficiency and scalability analysis

applications necessitate consideration of the road network. Incorporating the road network structure into multi-modal trajectory representation and obtaining generic representations suitable for road network-based tasks remain challenges that we aim to address in future work.

6 Conclusion

In this paper, we propose a generic contrastive learning-based multi-modal trajectory representation learning framework, namely CLMTR. CLMTR incorporates intra- and inter-trajectory contrastive learning components. These components collectively capture the correlations among diverse modal features within a trajectory and the relationships among different trajectories, generating semantically enriched and generic trajectory representations that can be applied to various downstream tasks. CLMTR also involves multi-modal feature embedding methods coupled with an attention-based fusion method, capturing the multi-modal characteristics and adaptively obtaining the unified embeddings. We conduct extensive experiments on two real-world trajectory datasets for three downstream tasks. The experimental results demonstrate the superior performance of CLMTR compared with the state-of-the-art approaches.

Funding This work was supported by the Oceanic Interdisciplinary Program of Shanghai Jiao Tong University (SL2023ZD102), the Nation Natural Science Foundation of China (62302294), and Alibaba Group through Alibaba Innovative Research (AIR) Program.

Data Availability The datasets analysed during the current study are available in: Geolife [43] and T-Drive [44].

Declarations

Conflict of Interest The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

1. Wang S, Bao Z, Culpepper JS, Sellis T, Sanderson M, Qin X (2017) Answering top-k exemplar trajectory queries. In: ICDE
2. Wang S, Bao Z, Culpepper JS, Cong G (2021) A survey on trajectory data management, analytics, and learning. CSUR 54(2):1–36
3. Wang S, Bao Z, Culpepper JS, Sellis T, Qin X (2019) Fast large-scale trajectory clustering. VLDB 13(1):29–42
4. Su H, Liu S, Zheng B, Zhou X, Zheng K (2020) A survey of trajectory distance measures and performance evaluation. VLDBJ 29:3–32
5. Rao X, Chen L, Liu Y, Shang S, Yao B, Han P (2022) Graph-flashback network for next location recommendation. In: SIGKDD
6. Jing Q, Liu S, Fan X, Li J, Yao D, Wang B, Bi J (2022) Can adversarial training benefit trajectory representation? an investigation on robustness for trajectory similarity computation. In: CIKM
7. Li X, Zhao K, Cong G, Jensen CS, Wei W (2018) Deep representation learning for trajectory similarity computation. In: ICDE
8. Yao D, Cong G, Zhang C, Bi J (2019) Computing trajectory similarity in linear time: a generic seed-guided neural metric learning approach. In: ICDE
9. Yao D, Hu H, Du L, Cong G, Han S, Bi J (2022) Trajgat: a graph-based long-term dependency modeling approach for trajectory similarity computation. In: SIGKDD

10. Yao D, Zhang C, Zhu Z, Huang J, Bi J (2017) Trajectory clustering via deep representation learning. In: IJCNN
11. Zhang Y, Liu A, Liu G, Li Z, Li Q (2019) Deep representation learning of activity trajectory similarity computation. In: ICWS
12. Yang P, Wang H, Zhang Y, Qin L, Zhang W, Lin X (2021) T3s: effective representation learning for trajectory similarity computation. In: ICDE
13. Fang Z, Du Y, Chen L, Hu Y, Gao Y, Chen G (2021) E 2 dtc: an end to end deep trajectory clustering framework via self-training. In: ICDE
14. Yue M, Li Y, Yang H, Ahuja R, Chiang YY, Shahabi C (2019) Detect: deep trajectory clustering for mobility-behavior analysis. In: Big Data
15. Zheng B, Yuan NJ, Zheng K, Xie X, Sadiq S, Zhou X (2015) Approximate keyword search in semantic trajectory database. In: ICDE
16. Zheng K, Zheng B, Xu J, Liu G, Liu A, Li Z (2017) Popularity-aware spatial keyword search on activity trajectories. World Wide Web 20:749–773
17. Song X, Xu J, Zhou R, Liu C, Zheng K, Zhao P, Falkner N (2020) Collective spatial keyword search on activity trajectories. GeoInformatica 24:61–84
18. Furtado AS, Kopanaki D, Alvares LO, Bogorny V (2016) Multidimensional similarity measuring for semantic trajectories. Trans GIS 20(2):280–298
19. Petry LM, Ferrero CA, Alvares LO, Renso C, Bogorny V (2019) Towards semantic-aware multiple-aspect trajectory similarity measuring. Trans GIS 23(5):960–975
20. Li Y, Fu K, Wang Z, Shahabi C, Ye J, Liu Y (2018) Multi-task representation learning for travel time estimation. In: SIGKDD
21. Han J, Liu H, Liu S, Chen X, Tan N, Chai H, Xiong H (2023) ieta: a robust and scalable incremental learning framework for time-of-arrival estimation. In: SIGKDD
22. Zhang X, Zhao W (2023) Deep normalization cross-modal retrieval for trajectory and image matching. In: DASFAA
23. Liu A, Zhang Y, Zhang X, Liu G, Zhang Y, Li Z, Zhao L, Li Q, Zhou X (2020) Representation learning with multi-level attention for activity trajectory similarity computation. TKDE 34(5):2387–2400
24. Zhang H, Zhang X, Jiang Q, Zheng B, Sun Z, Sun W, Wang C (2020) Trajectory similarity learning with auxiliary supervision and optimal matching. In: IJCAI
25. Yang P, Wang H, Lian D, Zhang Y, Qin L, Zhang W (2022) Tmn: trajectory matching networks for predicting similarity. In: ICDE
26. Han P, Wang J, Yao D, Shang S, Zhang X (2021) A graph-based approach for trajectory similarity computation in spatial networks. In: SIGKDD
27. Zhou S, Han P, Yao D, Chen L, Zhang X (2022) Spatial-temporal fusion graph framework for trajectory similarity computation. World Wide Web, 1–23
28. Fu TY, Lee WC (2020) Trembr: exploring road networks for trajectory representation learning. TIST 11(1):1–25
29. Fang Z, Du Y, Zhu X, Hu D, Chen L, Gao Y, Jensen CS (2022) Spatio-temporal trajectory similarity learning in road networks. In: SIGKDD
30. He K, Fan H, Wu Y, Xie S, Girshick R (2020) Momentum contrast for unsupervised visual representation learning. In: CVPR
31. Chen T, Kornblith S, Norouzi M, Hinton G (2020) A simple framework for contrastive learning of visual representations. In: ICML
32. Babaev D, Ovsov N, Kireev I, Ivanova M, Gusev G, Nazarov I, Tuzhilin A (2022) Coles: contrastive learning for event sequences with self-supervision. In: SIGMOD
33. Gao T, Yao X, Chen D (2021) Simcse: simple contrastive learning of sentence embeddings. In: EMNLP
34. Giorgi J, Nitski O, Wang B, Bader G (2020) Declutr: deep contrastive learning for unsupervised textual representations. In: ACL
35. Zhou F, Wang P, Xu X, Tai W, Trajcevski G (2021) Contrastive trajectory learning for tour recommendation. ACM Transactions on Intelligent Systems and Technology (TIST) 13(1):1–25
36. Deng L, Zhao Y, Fu Z, Sun H, Liu S, Zheng K (2022) Efficient trajectory similarity computation with contrastive learning. In: CIKM, pp. 365–374
37. Liu X, Tan X, Guo Y, Chen Y, Zhang Z (2022) Cstrm: contrastive self-supervised trajectory representation model for trajectory similarity computation. Comput Commun 185:159–167
38. Chang Y, Qi J, Liang Y, Tanin E (2023) Contrastive trajectory similarity learning with dual-feature attention. In: ICDE
39. Jiang J, Pan D, Ren H, Jiang X, Li C, Wang J (2023) Self-supervised trajectory representation learning with temporal regularities and travel semantics. In: ICDE

40. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser Ł, Polosukhin I (2017)(2017) Attention is all you need. Advances in neural information processing systems. 30
41. Douglas, D.H., Peucker, T.K.: Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. Cartographica: the international journal for geographic information and geovisualization 10(2), 112–122 (1973)
42. Oord Avd, Li Y, Vinyals O (2018) Representation learning with contrastive predictive coding. arXiv preprint arXiv:1807.03748
43. Zheng Y, Xie X, Ma W-Y et al (2010) Geolife: a collaborative social networking service among user, location and trajectory. IEEE Data Eng Bull 33(2):32–39
44. Yuan J, Zheng Y, Zhang C, Xie W, Xie X, Sun G, Huang Y (2010) T-drive: driving directions based on taxi trajectories. In: SIGSPATIAL